In [1]:
```
pip install nltk
```

```
Requirement already satisfied: nltk in ./anaconda3/lib/python3.10/site-packages (3.8.1)
Requirement already satisfied: regex>=2021.8.3 in ./anaconda3/lib/python3.10/site-packages
(from nltk) (2022.7.9)
Requirement already satisfied: tqdm in ./anaconda3/lib/python3.10/site-packages (from nltk)
(4.64.1)
Requirement already satisfied: click in ./anaconda3/lib/python3.10/site-packages (from nltk)
(8.0.4)
Requirement already satisfied: joblib in ./anaconda3/lib/python3.10/site-packages (from nlt
k) (1.1.1)
Note: you may need to restart the kernel to use updated packages.
```

In [2]:
```
import nltk
nltk.download("punkt")
nltk.download("stopwords")
nltk.download('wordnet')
nltk.download('averaged_perceptron_tagger')
```

```
[nltk_data] Downloading package punkt to /home/student/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     /home/student/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /home/student/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /home/student/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
```

Out[2]:
```
True
```

In [3]:
```
text= """Tokenization is the first step in text analytics. The
process of breaking down a text paragraph into smaller chunks
such as words or sentences is called Tokenization."""
```

In [4]:
```
from nltk.tokenize import sent_tokenize
tokenized_text=sent_tokenize(text)
print(tokenized_text)
```

```
['Tokenization is the first step in text analytics.', 'The\nprocess of breaking down a text
paragraph into smaller chunks\nsuch as words or sentences is called Tokenization.']
```

In [5]:
```
from nltk.tokenize import word_tokenize
tokenized_word=word_tokenize(text)
print(tokenized_word)
```

```
['Tokenization', 'is', 'the', 'first', 'step', 'in', 'text', 'analytics', '.', 'The', 'proce
ss', 'of', 'breaking', 'down', 'a', 'text', 'paragraph', 'into', 'smaller', 'chunks', 'such
', 'as', 'words', 'or', 'sentences', 'is', 'called', 'Tokenization', '.']
```

In [6]:
```
from nltk.corpus import stopwords
stop_words=set(stopwords.words("english"))
print(stop_words)
```

```
{"you're", 'their', 'has', 'not', 'it', "should've", 'ours', 'about', 'there', 'where', 'o',
'nor', 'haven', "doesn't", "weren't", "aren't", 'me', 'into', "hasn't", 'mustn', 'until', 'o
n', 'now', 'again', 'does', "don't", "you'd", 'him', 'out', 'being', 'can', 'further', 'thei
rs', 'which', "wasn't", 'her', 'should', 'himself', 'weren', 'won', 'very', 'your', 'yoursel
ves', 'you', 'the', 'when', 'did', 'yourself', 'my', "that'll", 'mightn', 'any', 'between',
'each', "wouldn't", 'both', "mightn't", 'myself', 'wouldn', 'hers', 'below', 'do', 'such', '
no', 'doing', 'isn', 'themselves', 'down', "it's", "shouldn't", 're', 'to', 've', 'our', 'yo
urs', 'from', 'other', "hadn't", 'only', 'don', 'needn', 'as', 'off', 'these', 'and', 'shan
', 'herself', 'at', 'ma', 'up', 'above', "needn't", 'that', 'they', 'y', "won't", 'because',
'hasn', 'if', 'against', 'own', 'through', 'so', 'is', 'with', 'we', "haven't", 'was', 'have
', 'i', 'aren', 'm', "isn't", 'before', 'why', 'them', 'most', "couldn't", 'be', 'who', 'by
', "you'll", 'of', 'same', 'than', 'ourselves', 'itself', 'shouldn', "she's", 'wasn', 'whom
', 'after', 'he', 'she', 'over', 'under', 't', 'will', 'during', 'then', 'couldn', 'hadn', '
while', 'more', 'been', "didn't", 'having', 'here', 's', "you've", 'a', 'ain', 'few', 'what
', "shan't", 'had', "mustn't", 'some', 'those', 'in', 'an', 'd', 'were', 'too', 'its', 'didn
', 'but', 'am', 'once', 'this', 'or', 'all', 'are', 'for', 'his', 'just', 'll', 'doesn', 'ho
w'}
```

In [7]:

```
import re
text= "How to remove stop words with NLTK library in Python?"
text=re.sub('[^a-zA-Z]',' ',text)
tokens=word_tokenize(text.lower())
filtered_text=[]
for w in tokens:
    if w not in stop_words:
        filtered_text.append(w)
print("Tokenzied Sentence:",tokens)
print("Filterd Sentence:",filtered_text)
```

```
Tokenzied Sentence: ['how', 'to', 'remove', 'stop', 'words', 'with', 'nltk', 'library', 'in
', 'python']
Filterd Sentence: ['remove', 'stop', 'words', 'nltk', 'library', 'python']
```

In [8]:

```
from nltk.stem import PorterStemmer
e_words=["wait", "waiting", "waited", "waits"]
ps=PorterStemmer()
for w in e_words:
    rootWord=ps.stem(w)
print(rootWord)
```

```
wait
```

In [9]:

```
import nltk
from nltk.stem import WordNetLemmatizer
wordnet_lemmatizer=WordNetLemmatizer()
text = "studies studying cries cry"
tokenization=nltk.word_tokenize(text)
for w in tokenization:
    print("Lemma  for {} is {}".format(w,wordnet_lemmatizer.lemmatize(w)))
```

```
Lemma  for studies is study
Lemma  for studying is studying
Lemma  for cries is cry
Lemma  for cry is cry
```

In [13…

```
import nltk
from nltk.tokenize import word_tokenize
data="The pink sweater fit her perfectly"
words=word_tokenize(data)
for w in words:
    print(nltk.pos_tag([w]))
```

```
[('The', 'DT')]
[('pink', 'NN')]
[('sweater', 'NN')]
[('fit', 'NN')]
[('her', 'PRP$')]
[('perfectly', 'RB')]
```

In [14…

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
documentA = 'Jupiter is the largest Planet'
documentB = 'Mars is the fourth planet from the Sun'
```

In [18…

```
bagOfwordsA=documentA.split(' ')
bagOfwordsB=documentB.split(' ')
uniquewords=set(bagOfwordsA).union(set(bagOfwordsB))
numofwordsA=dict.fromkeys(uniquewords,0)
for words in bagOfwordsA:
    numofwordsA[words] += 1
numofwordsB=dict.fromkeys(uniquewords,0)
for words in bagOfwordsB:
    numofwordsB[words] += 1
```

In [20…

```
def computeTF(wordDict,bagOfWords):
    tfdict={}
    bagsofwordcount=len(bagOfWords)
    for word,count in wordDict.items():
        tfdict[word]=count/ float(bagsofwordcount)
    return tfdict
tfA=computeTF(numofwordsA,bagOfwordsA)
tfB=computeTF(numofwordsB,bagOfwordsB)
```

In [24…

```
def computeIDF(documents):
    import math
    N = len(documents)
    idfDict = dict.fromkeys(documents[0].keys(), 0)
    for document in documents:
        for word, val in document.items():
            if val > 0:
                idfDict[word] += 1
    for word, val in idfDict.items():
        idfDict[word] = math.log(N / float(val))
    return idfDict
idfs = computeIDF([numofwordsA, numofwordsB])
idfs
```

Out[24]:

```
{'the': 0.0,
 'Jupiter': 0.6931471805599453,
 'is': 0.0,
 'Sun': 0.6931471805599453,
 'from': 0.6931471805599453,
 'fourth': 0.6931471805599453,
 'Planet': 0.6931471805599453,
 'largest': 0.6931471805599453,
 'planet': 0.6931471805599453,
 'Mars': 0.6931471805599453}
```

In [26…

```
def computeTFIDF(tfBagOfWords, idfs):
    tfidf = {}
    for word, val in tfBagOfWords.items():
        tfidf[word] = val * idfs[word]
    return tfidf
tfidfA = computeTFIDF(tfA, idfs)
tfidfB = computeTFIDF(tfB, idfs)
df = pd.DataFrame([tfidfA, tfidfB])
df
```

Out[26]:

	the	Jupiter	is	Sun	from	fourth	Planet	largest	planet	Mars
0	0.0	0.138629	0.0	0.000000	0.000000	0.000000	0.138629	0.138629	0.000000	0.000000
1	0.0	0.000000	0.0	0.086643	0.086643	0.086643	0.000000	0.000000	0.086643	0.086643

In [ ]: